APPENDIX F

```
#include    <stdio.h>
#include    <unistd.h>
#include    <errno.h>
#include    <stdlib.h>
#include    <sys/types.h>
#include    <sys/stat.h>
#include    <fcntl.h>
#include    <string.h>
#include    "fastdb.h"

extern int  FalseDigs ;
extern int  LinksFollowed ;

extern "C" long random();

// fastdb query function
int QueryRecord ( char *key, char *record );

void usage( char *progname )
{
    printf (
        "Usage: %s [-s] [-l <loop count>] [-r <random key max>] <key(s) ...>\n",
        progname );
    printf ( "  [-s]                    - seed random number generator\n" );
    printf ( "  [-l <loop count>]       - number of queries to run\n" );
    printf ( "  [-r <random key max>] - max number of records in Data\n" );
    exit(1);
}

//
// test program
//
// The -s option will seed the random number generator to produce
// different set of random numbers evertime it is invoked.
//
// The -l option will cause the query program to loop the specified
// number of times, over the keys specified.
//
// The -r option will generate random 7 Hex digits keys designed to
// work the the Data file created by "mkdata".  For example:
//
//  getrec -l 10000 -r 1000000
//
// will query 10,000 time using random keys from 0 to 1,000,000
// in 7 Hex digit format.  It also checks the answer from the
// record found.
//
main( int argc, char *argv[] )
{
    char    record[MAXRECORD];
    char    *key;
    int i;
    int loop = 1;
    int rmax = 0;
    long    rval;
    long    rmask;
    char    rkey[10];
```

```
    int seed;
    int value;
    char    *ptr;

    while ((i = getopt(argc, argv, "?sl:r:")) != -1)
        switch (i)
            {
            case 's':
                seed = (unsigned int) time(0) * getpid();
                seed = seed & 0x3fff;
                printf ( "seed = %d\n", seed );
                srandom( seed );
                break;

            case 'l':
                loop = atoi(optarg);
                if (loop < 1) loop = 1;
                break;

            case 'r':
                rmax = atoi(optarg);
                if (rmax < 0) rmax = 0;
                break;

            default:
                usage(argv[0]);
            }

    // how many keys are there?
    int keycount = argc - optind;
    if (keycount < 1 && !rmax )
        usage(argv[0]);
    else if (keycount > 0 && rmax )
        usage(argv[0]);
    else if ( rmax )
        {
        // use one key
        keycount = 1;

        // compute the minimum bit mask for max random value
        for ( rmask = 0x3fffffff; rmask >= rmax; rmask = rmask >> 1 )
            /* keep shifting */ ;
        rmask = (rmask * 2) + 1;    // one too many
        printf ( "using rmask = 0x%8.8x\n", rmask );
        }

    while ( loop-- )
        {
        // find each key in Data base
        for (i = 0 ; i < keycount; i++)
            {
            if ( !rmax )
                key = argv[optind + i];
            else
                {
                while ( (rval = random() & rmask) >= rmax )
                    /* keep looking*/ ;
```

```
                sprintf ( rkey, "%7.7x", rval );
                key = rkey;
                }

            if ( QueryRecord ( key, record ) )
                {
                if ( rmax )
                    {
                    // check record for right value
                    ptr = strchr ( record, '\t' );
                    if ( !ptr )
                        {
                        printf ( "No tab in: %s\n",
                            record );
                        exit(1);
                        }
                    sscanf ( ptr, "%d", &value );
                    if ( rval != value )
                        {
                        printf ( "Bad record: %s\n",
                            record );
                        exit(1);
                        }
                    }
                if ( !loop )
                    printf ( "Found: %s\n", record );
                }
            else
                {
                printf ("Record '%s' not found.\n", key );
                }
            }
        }
    printf ( "FalseDigs = %d, LinksFollowed = %d\n", FalseDigs, LinksFollowed );
}
```